REMARKS

I. Summary of the Examiner's Action

A. Claim Rejections

As set forth in paragraph 2 of the July 27 Office Action, claims 12 – 15 stand rejected under 35 U.S.C. § 101 as being directed to non-statutory subject matter.

As set forth in paragraph 3 of the July 27 Office Action, claims 1 – 22 stand rejected under 35 U.S.C. § 112, second paragraph as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

As set forth in paragraph 6 of the July 27 Office Action, claims 1 – 3, 10 – 19, 21 and 22 stand rejected under 35 U.S.C. § 102(e) as being anticipated by United States Patent No. 6,938,149 to Kunimatsu *et al.* (hereinafter "the Kunimatsu patent").

As set forth in paragraph 16 of the July 27 Office Action, claims 4 – 9 and 20 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over the Kunimatsu patent.

These rejections are respectfully disagreed with, and are traversed below.

II. Applicant's Response

A. Rejection of Claims 12 – 15 under 35 U.S.C. § 101

Applicants have amended claim 12 thereby obviating the rejection of claims 12 – 15 on the basis of 35 U.S.C. § 101.

B. Rejection of Claims 1 - 22 under 35 U.S.C. § 112, second paragraph

Applicants have amended claim 1 and the remaining claims to more particularly point out and distinctly claim their invention. No new matter has been added by these amendments.

Support for amendments to claim 1 can be found at, for example FIGS. 4, 5; page 12, lines 1 – 4; page 13, line 1 – page 15, line 7; page 16, line 10 – page 17, line 6. In particular, the portion appearing at page 16, line 10 – page 17, line 6 indicates that the named architected register may be assigned a new physical register or a general system memory location as a result of a name level instruction.

Support for amendments to claim 2 – 9 can be found at page 16, line 10 – page 18, line 2. In particular, the amendments to claims 2 – 9 make clear that although the name level instruction may initiate the assignment of a new physical memory location to a named architected register upon receipt of a name level instruction changing the name level of the named architected register, how the assignment is performed depends entirely on the characteristics and functionality of the processor; *see*, for example, page 14, lines 18 – 20.

C. Rejection of Claims 1 – 3, 10 – 19, 21 and 22 under 35 U.S.C. § 102(e), second paragraph

Claim 1 (as amended) recites:

> 1. A method for managing registers in a processor, comprising:
>
> introducing a name level instruction for at least one of a named architected register into an instruction set, the instruction set for use with the processor;
>
> allowing a programmer to change the current name level of the at least one named architected register via said name level instruction, the at least one named architected register available to the programmer as an additional named architected register as a result of a name level change; and
>
> assigning a new physical memory location to the at least one named architected register upon receipt of the name level instruction instituting a name level change for the at least one named architected register.

It is not seen where the art relied upon by the Examiner, whether taken singly or in combination, has anything to do with the subject matter of claim 1. In Applicants' invention, programmers are provided with a new instruction for their use, a so-called *name level instruction.*

With the name level instruction, a programmer can change the name level of an architected register, causing the architected register to operate effectively as a new architected register. In processors with register renaming apparatus, name level instructions when used by programmers initiate register renaming operations. Although admittedly the name level instruction may initiate register renaming operations, the name level instruction is not concerned with register renaming *per se*; rather, it is concerned with avoiding the need for register spilling operations by adding a new architected register for use by a programmer when the number of architected registers established by an instruction set architecture has been exhausted.

The name level instruction avoids the need for spilling operations in certain processors by working in combination with register renaming apparatus of the processors. Nonetheless, it is the register renaming apparatus of these processors that assign a new physical memory location to the architected register when the name level of the architected register is changed, and not the name level instruction. The programmer thus avoids the need to program spilling operations when the renaming apparatus of the processor changes the physical memory location associated with the architected register in response to receipt of a name level instruction.

The operations of Applicants' invention are described with particularity at page 13, line 1 – page 14, line 14 of the Application reproduced here (emphasis added):

"The invention disclosed herein presents a new method of reducing the overhead of register management. One embodiment of the invention introduces a concept of name level for each of the named architected registers in the processor. When the machine begins execution, each of the register names has the level of '0' associated with it. The invention introduces a new instruction called 'nameLev', which allows the programmer to change the 'current' name level of a register name. Referring to FIG. 4, which shows an example of when a programmer runs out of usable registers after the use of R1. In this case, the programmer simply inserts the following nameLev instruction:

nameLev R1, 1

which associates the new name level of '1' with the architected register name R1. From that point on until the programmer changes the name level of R1 back to '0', a new register named R1 will be available for use. The program follows and uses the new R1, as needed. When the new R1 is no longer required or the 'old' register R1 must be used, the programmer inserts the instruction:

nameLev R1, 0

at that point in the program. From that point on, when a program instruction uses the name R1, it refers to R1 at name level 0. FIG. 5 shows what happens inside the processor when the example instruction sequence is presented for execution. When execution of this piece of code begins, the LOAD R1 instruction creates an internal name for R1 as R80. At the first use of R1, internal register R80 is used to supply the desired value in R1. An instruction then executes that requires an architected register name, but all names (in this 2-register machine example) are taken. The programmer then inserts the namelev R1, 1 instruction. The nameLev instance informs the machine that the programmer intends to use a new internal register for R1. The machine

> uses its internal name management method (in an out-of-order with renaming processor, a rename map table), to remember the change. The next time an instruction uses the name R1 as one of its source registers, the newly remembered internal register is used to supply the source operand for that instruction. On the other hand, if and when an instruction writes its results into R1 (not shown in this example), the results are written to a new physical register assigned by the hardware in the normal course of execution.

The provision of what is effectively an extra architected register thus avoids the need for unnecessary spill code.

Again, although the name level instruction can work in combination with register renaming apparatus to effectively provide one or more "new" architected registers for use by a programmer, the programmer is not using the name level instruction to "control" register renaming operations. In fact, Applicants' invention eliminates the need for a programmer to understand in detail the inner workings and limitations of a particular processor implementation and preserves the ability of a programmer to program solely with reference to architected registers, whether the architected registers are implemented by the instruction set architecture of the processor or whether the architected registers are added by the name level instruction. This is described at page 14, line 15 – page 15, line 7 of the Application (emphasis added):

> "To summarize, an extra named register was made available to the programmer on demand and when the use of the extra register was no longer needed, it was taken out of use. The method used the exact same encoding of

instructions as in FIG. 3, with the only additional overhead being imposed was that of the two extra instructions. The process of mapping the new name levels to internal resources was completely managed by the processor implementation and is completely hidden from the programmer. Thus the original goal of an architectural specification, which allows the programmer complete independence from the machine implementation while guaranteeing the correctness of program execution, is achieved. This method completely frees the programmer from having to study and analyze different trade-offs when spilling and re-filling registers. The method makes the task of register allocation, assignment, spilling, and re-filling completely redundant as the programmer is provided with a large number of name levels for the registers that he wishes to use."

The advantages of Applicants' invention are summarized at page 18, line 3 – page 19, line 5, reproduced here (emphasis added):

"Some of the advantages of the present invention are as follows. This invention makes more architected registers available to the programmer, via overloading or extending the register namespace dynamically under program control. At least as many as the number of physical registers in the processor are available for direct use by the programmer at low access latencies. Any program that uses more registers than that could experience performance penalties, but will still execute as expected by the programmer. In other words, the programmer now has the flexibility to trade-off performance for assured accuracy.

* * *

Further, the present invention reduces the power consumption in the processor by reducing the unnecessary traffic to the storage hierarchy, which is necessitated by the spill and re-fill instructions in processors without the present invention.

Further, improved performance of an application on a processor that uses the present invention is achieved by reducing unnecessary spill and re-fill instructions, thereby reducing the burden on the execution pipeline of the processor, the instruction memory hierarchy of the processor, the data memory hierarchy of the processor, and the cache coherence traffic in the system."

In contrast, the methods and apparatus of Kunimatsu are not concerned with effectively providing new architected registers to programmers when needed; rather, Kunimatsu is concerned mainly with register renaming operations, and to a limited degree, with providing users with some degree of control over the register renaming operations themselves, as stated at column 5, lines 17 - 30:

"As described above, in the present embodiment, the correspondence between the physical register number and the logical register number can be designated for each bundle with respect to arbitrary bundles using the register rename instruction as the dedicated instruction for designating the correspondence between the physical register number and the logical register number. Therefore, the physical register number can be selected so that a pipeline babble [sic] is not generated, and as a result a processing speed of the processor can be enhanced.

Moreover, the concrete content of the register rename instruction can arbitrarily be designated by a programmer, and the rename processing can therefore be performed if necessary."

Applicants' invention does not operate in this manner. In order to accurately program the instruction mentioned at column 5, lines 27 – 30 of Kunimatsu, the programmer must have detailed knowledge of the physical registers of the particular processor implementation. If not, the programmer may program a register renaming operation with non-existent physical registers.

In contrast, Applicants' invention avoids the need for such detailed knowledge of a processor implementation (see again page 14, lines 18 – 20). In addition, it is noted that the name level instruction as described and claimed does not recite subject matter for the precise purpose of controlling internal register renaming operations. All the name level instruction does is change the name level assigned to a named architected register. As stated previously, although use of a name level instruction may initiate register renaming operations in dependence on the particular implementation of the processor executing the instruction, the name level instruction itself does not control the register renaming operations themselves including, for example, what new physical memory location is assigned to the named architected register with the new name level.

For the foregoing reasons Applicants respectfully submit that claim 1 is patentable over the art of record, whether taken singly or in combination. Accordingly, Applicants respectfully request that the rejection of claim 1 be withdrawn. Applicants respectfully submit that independent claims 12 and 16 are patentable both for reasons similar to claim 1 and for reasons attributable to their independently-recited features. Applicants therefore respectfully request that the rejection of independent claims 12 and 16 be withdrawn as well. Further, applicants respectfully submit that dependent claims 2 – 3; 10 – 11; 13 – 15; and 21 – 22 are allowable both as depending from allowable base claims and for reasons attributable to their independently-recited features. Therefore, Applicants respectfully request that the rejection of dependent claims 2 – 3; 10 – 11; 13 – 15; and 21 – 22 be withdrawn as well.

Applicants have cancelled claims 17 – 19, thereby mooting the rejection of these claims.

D. Rejection of Claims 4 – 9 and 20 under 35 U.S.C. § 103(a)

Applicants respectfully submit that claims 4 – 9 and 20 are allowable both as depending from allowable base claims and for reasons attributable to their independently-recited features. Applicants therefore respectfully submit that the rejection of claims 4 – 9 and 20 be withdrawn.

E. New Claims 23 – 25

Applicants have added new claims 23 – 25. Support for new claim 23 – 25 is found throughout the application *see*, for example, FIGS. 4 – 5, 7; page 13, line 1 – page 15, line 7; page 16, line 10 – page 17, line 6.

III. Conclusion

The Applicants submit that in light of the foregoing remarks the application is now in condition for allowance. Applicants therefore respectfully request that the outstanding rejections be withdrawn and that the case be passed to issuance.

Respectfully submitted,

September 19, 2006 David M. O'Neill (35,304)

Date

David M. O'Neill (Reg. No. 35,304)

Customer No.: 29683
HARRINGTON & SMITH, LLP
4 Research Drive
Shelton, CT 06484-6212
Telephone: (203)925-9400
Facsimile: (203)944-0245
email: DOneill@hspatent.com

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

9/19/2006 Elaine F. Muan

Date

Name of Person Making Deposit